UNIVERSITEIT
GENT

LABO DATASTRUCTUREN

verslag

benjamin fraeyman

Inhoud

Labo 1 2
Opdracht: 2
Oplossing: 2
Labo 2 3
Opdracht: 3
Oplossing: 3
Labo 3 4
Opdracht: 4
Oplossing: 4
Labo 4 5
Opdracht: 5
Oplossing: 5
Labo 5 6
Opdracht: 6
Oplossing: 6
Selectionsort: 6
Insertionsort: 6
BubbleSort: 6
Labo 6 8
Opdracht: 8
Zoeken van de mediaan van ‘n’ mediabestanden via een median heap. 8
Oplossing: 8
Labo 7 9
Opdracht: 9
Oplossing: 9
Bronnen: 10

Labo 1

Opdracht:

Ontwikkel een class MediaObject, waarin je naast algemene bestandsinfo (bestandsgrootte, hoogte, breedte, filename) ook spatio-temporele informatie (lat,long,datum) omtrent een mediabesand kan opslaan.

Maak een functie om de MediaObjecten te sorteren op datum.

Oplossing:

Naam van de klasse in programma: MediaObject.cs

Eigenschappen van een mediaobject:

- Filename: de volledige naam van het object
- Size: de grootte van het object [Bytes]
- Datum: tijdstip van creatie
- Bmp: de eigenlijke afbeelding
- Een paar eigenschappen voor de kleuren
- Een paar eigenschappen mbt afstand

Openen van de filebrowser gebeurt met de knop BrowserButton.

De actie gekoppelt aan deze knop is: BrowserButton_Click. Bij deze actie openen we een folder en laden we alle bestanden in en maken we voor elk bestand een nieuw mediaobject aan.

Het sorteren van de datum gebeurdt met de knop SortByDateButton.

De actie hieraan gekoppeld is: SortByDateButton_Click. Wanneer we deze actie aanroepen vergelijken we elk mediaobject met elkaar tot we de recentste terugvinden.

Labo 2

Opdracht:

Schrijf een binarysearchmethode om een mediabestand van een bepaalde datum te zoeken. Analyseer de tijdscomplexiteit en zet dit grafisch uit voor toenemende n (met n = het aantal mediafiles in de folder). De te zoeken datum wordt geselecteerd via een DateTimePicker.

Oplossing:

Naam van de klasse in programma: BinarySearchMethod.cs

(BinarySearch)

Werking:

Als [(oudste foto nieuwer is dan te vergelijken datum) of (nieuwste foto ouder is dan te vergelijken datum)]. Dan weten we dat de lijst de foto niet bevat.

Indien dit niet zo is, dan bevat de lijst waarschijnlijk de datum en werken we als volgt:

Vergelijk de datum met het middenste object, indien de te zoeken datum lager ligt, dan verwijder je de bovenste helft van de lijst. (onderste helft indien de datum hoger ligt)

Daarna kijk je weer naar het midden van de resterende lijst en kijk je terug waar de datum moet liggen.

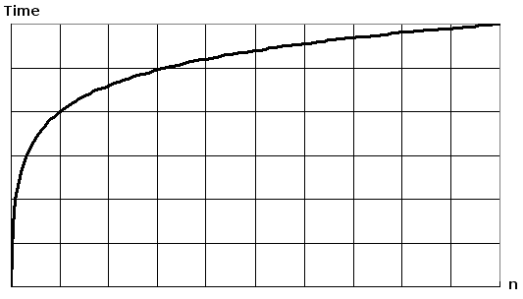
Dit doe je tot je de overeenkomende datum vind.

Tijdscomplexiteit:

Je kan log n (grondtal 2) keer doormidden delen en dan heb je een stuk van lengte 1.

Dus O (log n) stappen om te zoeken in een “gesorteerde” array.

Grafisch:



Labo 3

Opdracht:

Werk de binary search methode verder uit, zodat alle elementen worden teruggegeven die aan de zoekopdracht voldoen. Doe dit zo efficient mogelijk. Bespreek de impact hiervan op de complexiteit.

Oplossing:

Naam van de klasse in programma: BinarySearchMethod.cs

(BinarySearchExtra)

Werking:

Het vinden van het object is idem als in labo2.

Wanneer het object gevonden is dan kijk je in de laatst bekomen lijst of het links en rechtse object ook dezelfde datum hebben. Indien dit zo is, dan kijk je of de tweede linkse en rechtse ook dezelfde waarde hebben, enz.

Doe dit tot 1 van de 2 waardes (of alletwee) verschillen van het te zoeken object.

Indien 1 van de 2 waardes niet meer overeenkomt blijf je verder kijken aan de andere kant van de lijst tot die ook niet meer overeenkomt.

Wanneer beide waardes niet meer overeenkomen, heb je het interval van foto´s met dezelfde waarde.

Tijdscomplexiteit:

Daar het vergelijken van een waarde per bestand is dit in het slechtse geval voor alle waarden gelijk, dat betekent dus dat alle afbeeldingen doorlopen worden. Hierdoor wordt de complexiteit n*log(n).

Een verdere verbetering kan misschien zijn door weer binary search toe te passen op het laatste interval waar je in bent gaan zoeken. Hierdoor wordt de complexiteit: log(n) * log(n) * log(n). want je hebt je originele log(n), dan heb je weer een log(n) voor boven en onder.

Labo 4

Opdracht:

Voeg GPS geodata toe aan je MediaObjecten en lees deze data uit de file properties.

Oplossing:

Geodata wordt ingeladen bij het klikken op de knop “GeoDataButton”.

Dit wordt gedaan door de exiflibrary.

Onze geodata bevat:

- Longitude
- Latitude

Labo 5

Opdracht:

Sorteren van n mediabestanden op afstand tot huidige bestemming. Sorteer op basis van selection, insertion of bubbleSort.

Oplossing:

Gebruikte klasse in code: SortingMethods.cs

Selectionsort:

Werking:

1. Zoek de kleinste waarde in de lijst.
2. Verwissel het met de eerste waarde in de lijst.
3. Herhaal de bovenstaande stappen met de rest van de lijst.

Tijdscomplexiteit:

Zoeken van minimum: $1 + 2 + \ldots + (n-1) + n = (n + 1)\ (n/2) = n^2/2 + n/2 = n^2$

Insertionsort:

Werking:

Het begint door de eerste twee elementen uit de set te sorteren. Als deze op hun plaats staan, voegen we het derde element op de juiste plaats toe. Dit doen we totdat we alle elementen op hun plaats hebben gezet.

Tijdscomplexiteit:

De tijdscomplexiteit is $O(n^2)$ in de meeste gevallen en in het beste geval, als de waarden al bijna gesorteerd zijn, is de tijdscomplexiteit $O(n)$.

BubbleSort:

Werking:

1. Loop door de te sorteren rij van n elementen en vergelijk elk element met het volgende. Verwissel beide als ze in de verkeerde volgorde staan. Schuif dan een stapje op.

2. Loop opnieuw door de rij, maar ga nu door tot het voorlaatste element, omdat het laatste element het grootste in de rij was.

3. Nog een keer, maar negeer dan de twee laatste elementen.

4. Ga zo door.

n. Nog een keer, maar negeer dan de laatste n-1 getallen.

n+1. Klaar.

Tijdscomplexiteit:

2 for-loops die beide doorlopen worden , dus n * n:

De tijdscomplexiteit is $O(n^2)$.

Performantie:

Test Case	Selection sort	Insertion sort	Bubble sort
Kleine dataset	n^2	n	n
Grote dataset	n^2	n^2	n^2
Willekeurige dataset	n^2	n^2	n^2
Gedeeltelijk gesorteerd	n^2	n	n^2
Dubbele waarden	n^2	n^2	n^2

Labo 6

Opdracht:

Zoeken van de mediaan van ‘n’ mediabestanden via een median heap.

Oplossing:

Gebruikte klasse in code: HEAPSS.cs

Er is nood aan 2 heaps, 1 minheap en 1 maxheap. Elk element in de minheap is >= de mediaan en elk element in de maxheap is <= de mediaan.

Wanneer de minheap 1 element meer bevat dan de maxheap, dan is de top van de minheap de mediaan. Wanneer de maxheap 1 element meer bevat dan de minheap, dan is de top van de maxheap de mediaan.

Indien beide heaps even groot zijn, dan kun je de mediaan bepalen door het gemiddelde van beide toppen.

Wanneer het aantal elementen met meer dan 1 verschilt in beide heaps dan moet je het min/max van de heap nemen en deze in de andere heap steken.

(In de code is het makkelijker om eerst de mediaan te bepalen en aan de hand daarvan de min en maxheap op te stellen. Balanceren is dan niet nodig want we weten al waar de mediaan ligt en hoeveel elementen er in de heaps zullen zitten.)

Daar we weten dat een foto onder of overbelicht kan zijn, is de beste foto de mediaan.

Labo 7

Opdracht:

Bouw een hashtabel op met keywords van de MediaCollectie folder (dus niet per afbeelding, maar op folder-niveau). De gebruiker kan vrij enkele keywords opgeven (restrictie: elk keyword is 1 woord – voorzie controle). Schrijf je eigen hashfunctie!

Oplossing:

Gebruikt klase in code: Hashing.cs

We hashen de woorden door de hashfunctie: CalculateHash, we tellen op en vermenigvuldigen door een getal die maakt dat elk woord zijn eigen key heeft.

Opslaan gebeurt in HashItBro. Indien er al een bestand in hjet pad zit dan verwijderen we het en maken we een nieuwe.

Inlezen gebeurt in DeHashItBro.

Bronnen:

Figuur1

Big O selection sort

Bubble Sort

Big O